

## Lab Exercise 8

Dr. Sarvar Abdullaev  
s.abdullaev@inha.uz

April 5, 2019

The purpose of this lab is to write a simple blog application using Laravel framework by using Views, Routes, Controllers and some Facade Classes such as Store.

### 1 Clone Project and Install Dependencies

Follow below steps in order to ensure that your computer is set up to develop using Laravel framework:

1. Set `PATH='...;C:\xampp\php'` and in terminal type `php -v`. If it shows the version number, then your PHP pre-processor is accessible.
2. Download and install Composer here: <https://getcomposer.org/Composer-Setup.exe>. After installing, check in your terminal command `composer -v`.
3. Accept following lab assignment in Github Classroom: <https://classroom.github.com/a/UdoGPwP1> and clone newly created repository in your usual labs folder.
4. Open terminal inside that folder and run following command `composer install` to install all PHP dependencies of cloned project
5. Rename `.env.example` file to `.env` file. In command line run `mv .env.example .env` (Linux or MacOS) or `ren .env.example .env` (Windows)
6. Run following command afterwards: `php artisan key:generate`
7. Once all dependencies are installed, run following command `php artisan serve`. This will start a Laravel's own development web server at `http://localhost:8000`. Open it in your browser. You should be able to see Figure 1 web page:

### 2 Views and Blade Templating

In this exercise, you will create different parts of our blog application.

#### 2.1 Blog Section

Complete following tasks:

1. Create `blog` folder inside `/resources/views/` folder
2. Create 2 files inside `blog` folder: `index.blade.php` and `post.blade.php`
3. Copy the contents of `welcome.blade.php` to `blog\index.blade.php`
4. In `/routes/web.php` change root path from `welcome.blade.php` to `blog\index.blade.php`.
5. In `post.blade.php` extend `/layouts/master.blade.php` and write some static text in `content` section, so it looks as shown in Figure 2.

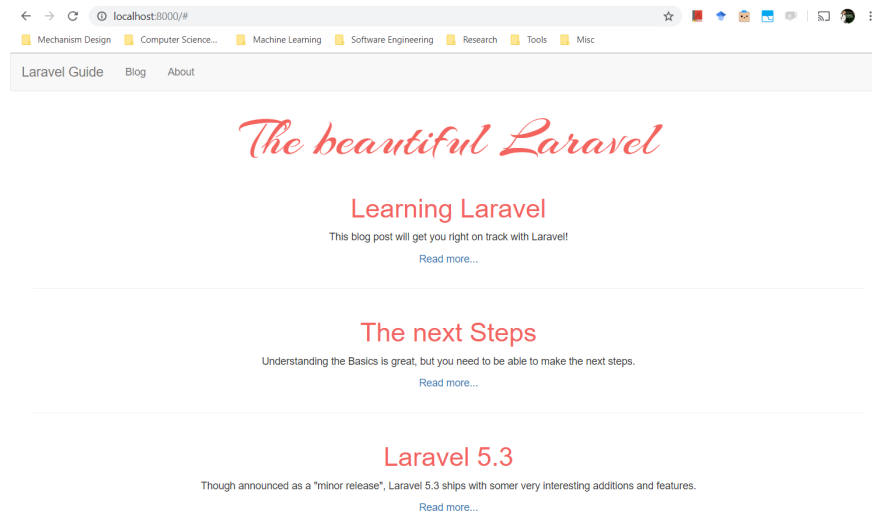


Figure 1: First View

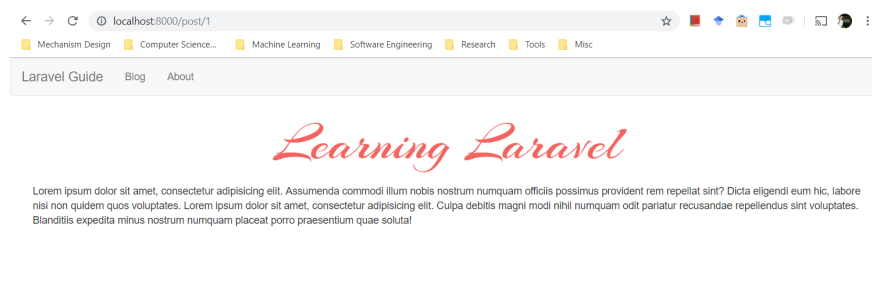


Figure 2: Post Detail View

6. As soon as you create a new template, set a route to it in `/routes/web.php`. Following URL can be used to display details of the post: `/post/{id}`
7. Modify `blog\index.blade.php` correspondingly so each 'Read more...' link refers to corresponding URL. For ex, post number 2 should refer to `/post/2`. You should use `route('url_name', ['id'=>$id])` function to generate corresponding URLs for detailed posts.

## 2.2 Admin Section

Complete following tasks:

1. Create `admin` folder inside `/resources/views/` folder
2. Create 3 files inside `admin` folder: `index.blade.php`, `create.blade.php` and `edit.blade.php` and create corresponding routes in `/routes/web.php` respectively: `admin/`, `admin/create` and `admin/edit/{id}`. Group these routes with prefix `admin`. Name above routes as `admin.index`, `admin.create`, `admin.edit`
3. Create `admin.blade.php` file inside `layouts` folder and copy the contents of `/layouts/master.blade.php`.
4. Create `admin-header.blade.php` inside `partials` folder and copy the contents of `header.blade.php`
5. Modify `partials\admin-header.blade.php`, so it contains link 'Posts' referring to route `admin.index`.
6. Replace reference to `header.blade.php` to `admin-header.blade.php` inside `/layouts/admin.blade.php`.

7. Modify `index.blade.php`, `create.blade.php` and `edit.blade.php`, so they all extend `/layouts/admin.blade.php`
8. Modify `index.blade.php`, so it contains 'New' button and a static post with some 'Edit' link. It should look as shown in Figure 3. Note you can use Bootstrap link button such as `<a class="btn btn-success">New Post</a>` for that. Check <https://getbootstrap.com/docs/4.3/components/buttons/> for more information.

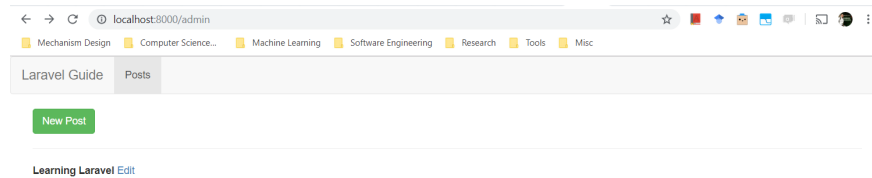


Figure 3: Admin Panel

9. Modify `create.blade.php`, so it contains 2 input fields 'Title' and 'Content' along with submit button. You can use Bootstrap forms here. After creating this form, change href of 'New' button `index.blade.php`, so when it is clicked, it refers to route `admin.create`.
10. Copy the contents of `create.blade.php` to `edit.blade.php`. Final form should look as shown in Figure 4.

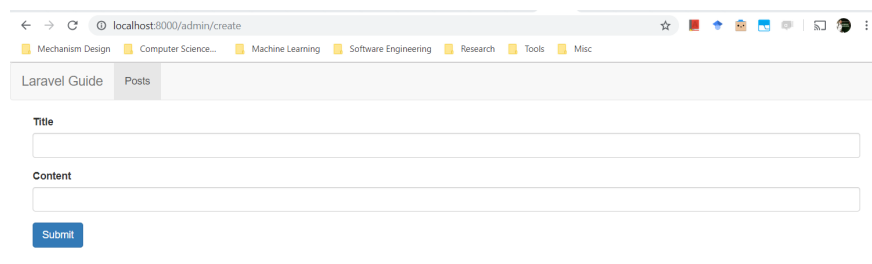


Figure 4: New Post Form

## 2.3 About Page

1. Create `/resources/views/other` folder and then create `about.blade.php` inside it. Create corresponding route `/about` inside `/routes/web.php`.
2. Modify `about.blade.php`, so it looks as shown in Figure 5

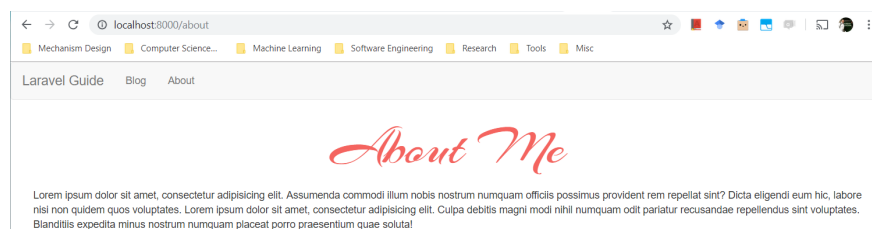


Figure 5: About Page

3. Modify `header.blade.php` so it refers correctly to `/about` route.

### 3 Working with Requests

In this section, you work with Requests and Validators and use Dependency Injection to obtain instances of corresponding request and validator objects.

1. Create `partials/errors.blade.php` reusable template which is going to display errors of validation. Write following snippet inside it:

```
@if(count($errors->all()))
    <div class="row">
        <div class="col-md-12">
            <div class="alert alert-danger">
                <ul>
                    @foreach($errors->all() as $error)
                        <li>{{ $error }}</li>
                    @endforeach
                </ul>
            </div>
        </div>
    </div>
@endif
```

As you can see it lists all errors passed to the template.

2. Go to `admin/index.blade.php` and make following modifications, so the final version look as shown below:

```
@extends('layouts.admin')

@section('content')
    @if(Session::has('info'))
        <div class="row">
            <div class="col-md-12">
                <p class="alert alert-info">{{ Session::get('info') }}</p>
            </div>
        </div>
    @endif

    <div class="row">
        <div class="col-md-12">
            <a href="{{ route('admin.create') }}" class="btn btn-success">New Post</a>
        </div>
    </div>
    <hr>
    <div class="row">
        <div class="col-md-12">
            <p><strong>Learning Laravel</strong>
            <a href="{{ route('admin.edit', ['id' => 1]) }}">Edit</a></p>
        </div>
    </div>
@endsection
```

Pay attention to how template engine check if there is a `info` variable stored in user session, and if yes, it shows that `info` as an alert.

3. Go to `admin/create.blade.php` and make following modifications, so the final version look as shown below:

```
@extends('layouts.admin')

@section('content')
    @include('partials.errors')
    <div class="row">
        <div class="col-md-12">
            <form action="{{ route('admin.create') }}" method="post">
                <div class="form-group">
                    <label for="title">Title</label>
                </div>
            </form>
        </div>
    </div>
```

```

        <input type="text" class="form-control" id="title" name="title">
    </div>
    <div class="form-group">
        <label for="content">Content</label>
        <input type="text" class="form-control" id="content" name="content">
    </div>
    @csrf
    <button type="submit" class="btn btn-primary">Submit</button>
</form>
</div>
</div>
@endsection

```

Don't forget to include CSRF field inside the form.

- Go to `/routes/web.php`, and create a new route to process POST request coming from above form.

```

Route::post('create', function( \Illuminate\Http\Request $request,
                                \Illuminate\Validation\Factory $validator) {
    $validation = $validator->make($request->all(), [
        'title' => 'required|min:5',
        'content' => 'required|min:10'
    ]);
    if ($validation->fails()) {
        return redirect()->back()->withErrors($validation);
    }
    return redirect()
        ->route('admin.index')
        ->with('info', 'Post created, Title: ' . $request->input('title'));
})->name('admin.create');

```

In above code snippet, route handler is accepting `$request` and `$validator` objects from Dependency Injector. They are registered inside `config/app.php` file under ServiceProviders. Handler validates the user input, and if everything is valid, it routes user to `admin.index` page with corresponding one-off Session variable `info` which is populated with user input. If user input contains a validation error it goes back to `admin.create` form with validation errors. You can learn about validation in Laravel here: <https://laravel.com/docs/5.8/validation>

- Go to `admin/edit.blade.php` and make following modifications, so the final version look as shown below:

```

@extends('layouts.admin')

@section('content')
    @include('partials.errors')
    <div class="row">
        <div class="col-md-12">
            <form action="{{ route('admin.update') }}" method="post">
                <div class="form-group">
                    <label for="title">Title</label>
                    <input
                        type="text"
                        class="form-control"
                        id="title"
                        name="title"
                        value="{{ $post['title'] }}">
                </div>
                <div class="form-group">
                    <label for="content">Content</label>
                    <input
                        type="text"
                        class="form-control"
                        id="content"
                        name="content"
                        value="{{ $post['content'] }}">
                </div>
                @csrf
                <button type="submit" class="btn btn-primary">Submit</button>
            </form>
        </div>
    </div>

```

```

    </div>
@endsection

```

6. Go to `/routes/web.php`, and create a new route to process POST request coming from above form.

```

Route::post('edit', function(\Illuminate\Http\Request $request,
                             \Illuminate\Validation\Factory $validator) {
    $validation = $validator->make($request->all(), [
        'title' => 'required|min:5',
        'content' => 'required|min:10'
    ]);
    if ($validation->fails()) {
        return redirect()->back()->withErrors($validation);
    }
    return redirect()
        ->route('admin.index')
        ->with('info', 'Post edited, new Title: ' . $request->input('title'));
})->name('admin.update');

```

## 4 Using Controller

In this section, you will move all your implemented functionality from `/routes/web.php` to controllers and write some additional functionality. Complete following steps:

1. Create `app/Post.php` file and define a `Post` class there. It should look as shown below:

```

<?php

namespace App;

use Illuminate\Session\Store;

class Post
{
    private $session;

    //asking session variable from Dep Inj.
    public function __construct(Store $session){
        $this->session=$session;
        $this->createDummyData();
    }

    public function getPosts()
    {
        return $this->session->get('posts');
    }

    public function getPost($id)
    {
        return $this->session->get('posts')[$id];
    }

    public function addPost($title, $content)
    {
        $posts = $this->session->get('posts');
        array_push($posts, ['title' => $title, 'content' => $content]);
        $this->session->put('posts', $posts);
    }

    public function editPost($id, $title, $content)
    {
        $posts = $this->session->get('posts');
        $posts[$id] = ['title' => $title, 'content' => $content];
        $this->session->put('posts', $posts);
    }
}

```

```

private function createDummyData()
{
    $posts = [
        [
            'title' => 'Learning Laravel',
            'content' => 'This blog post will get you right on track with Laravel!'
        ],
        [
            'title' => 'Something else',
            'content' => 'Some other content'
        ]
    ];
    $this->session->put('posts', $posts);
}
}

```

2. Go to terminal and run command `php artisan make:controller PostController`. This will create and register a new controller in `app\Http\Controllers`. Ensure that it contains all user actions implemented as follows:

```

<?php

namespace App\Http\Controllers;

use App\Post;
use Illuminate\Http\Request;

use App\Http\Requests;
use Illuminate\Session\Store;

class PostController extends Controller
{
    public function getIndex()
    {
        //explicitly asking Dependency Injector
        //to give new instance of Post
        $post = resolve('App\Post');
        $posts = $post->getPosts();
        return view('blog.index', ['posts' => $posts]);
    }

    public function getAdminIndex()
    {
        $post = resolve('App\Post');
        $posts = $post->getPosts();
        return view('admin.index', ['posts' => $posts]);
    }

    public function getPost($id)
    {
        $post = resolve('App\Post');
        $post = $post->getPost($id);
        return view('blog.post', ['post' => $post]);
    }

    public function getAdminCreate()
    {
        return view('admin.create');
    }

    public function getAdminEdit($id)
    {
        $post = resolve('App\Post');
        $post = $post->getPost($id);
        return view('admin.edit', ['post' => $post, 'postId' => $id]);
    }
}

```

```

public function postAdminCreate(Request $request)
{
    $this->validate($request, [
        'title' => 'required|min:5',
        'content' => 'required|min:10'
    ]);
    $post = resolve('App\Post');
    $post->addPost($request->input('title'),
        $request->input('content'));

    return redirect()
        ->route('admin.index')
        ->with('info', 'Post created, Title is: ' . $request->input('title'));
}

public function postAdminUpdate(Request $request)
{
    $this->validate($request, [
        'title' => 'required|min:5',
        'content' => 'required|min:10'
    ]);
    $post = resolve('App\Post');
    $post->editPost($request->input('id'),
        $request->input('title'),
        $request->input('content'));

    return redirect()
        ->route('admin.index')
        ->with('info', 'Post edited, new Title is: ' . $request->input('title'));
}
}

```

3. Go to `/routes/web.php`, and map all admin related routes to controller actions as shown below:

```

Route::group(['prefix' => 'admin'], function() {
    Route::get('', [
        'uses' => 'PostController@getAdminIndex',
        'as' => 'admin.index'
    ]);

    Route::get('create', [
        'uses' => 'PostController@getAdminCreate',
        'as' => 'admin.create'
    ]);

    Route::post('create', [
        'uses' => 'PostController@postAdminCreate',
        'as' => 'admin.create'
    ]);

    Route::get('edit/{id}', [
        'uses' => 'PostController@getAdminEdit',
        'as' => 'admin.edit'
    ]);

    Route::post('edit', [
        'uses' => 'PostController@postAdminUpdate',
        'as' => 'admin.update'
    ]);
});

```

4. Go to `blog\index.blade.php` and ensure that all posts are taken from `$posts` variable.

```

@extends('layouts.master')

@section('content')
    <div class="row">
        <div class="col-md-12">
            <p class="quote">The beautiful Laravel</p>
        </div>
    </div>

```



```

@foreach($posts as $post)
<div class="row">
    <div class="col-md-12 text-center">
        <h1 class="post-title">{{ $post['title'] }}</h1>
        <p>{{ $post['content'] }}!</p>
        <p><a href="{{ route('blog.post', ['id' => array_search($post, $posts)]) }}">Read more...</a>
    </div>
</div>
<hr>
@endforeach
@endsection

```

5. Go to `/routes/web.php`, and modify routes related to blog views as follows:

```

Route::get('/', [
    'uses' => 'PostController@getIndex',
    'as' => 'blog.index'
]);

Route::get('post/{id}', [
    'uses' => 'PostController@getPost',
    'as' => 'blog.post'
]);

```

6. Go to `admin/edit.blade.php` and add `<input type="hidden" name="id" value="{{ $postId }}">` field to the form, so it also passes the ID of the posted blog.

## 5 Final Solution

You can compare your solution with the final solution here: <https://github.com/iuthub/ip2019-lab-9/tree/solution>